Framework for Fast Prototyping of Applications Running on Reconfigurable System on Chip

Jan Viktorin, Pavol Korcek, Vlastimil Kosar and Jan Korenek
Brno University of Technology
Faculty of Information Technology
Bozetechova 1/2, 612 66 Brno, Czech Republic
Email: {iviktorin, ikorcek, ikosar, korenek}@fit.vutbr.cz

***Abstract*—Recently introduced chips with ARM based processors and programmable logic provide huge potential for digital signal processing, networking and other applications. Many IP cores and operating systems have been prepared for these chips to simplify the development process. Nevertheless, the integration of IP cores and operating system is not covered by any development tool yet. Developers have to design, implement and debug the communication between hardware and software part of the application. Therefore we propose Reconfigurable System on Chip (RSoC) Framework to support rapid prototyping of applications running on FPGA chips with a processor. The framework consists of FPGA logic and OS drivers to support communication between application core in the FPGA and application software on the host processor. Moreover, the framework allows to configure automatically address space of components in the FPGA and supports dynamic loading of drivers according to the FPGA configuration. The developer can focus only on the application software and accelerating core. For demonstration purposes, the framework is exploited in the example of a video processing application, where an image filter is running in the software and than is accelerated in the FPGA.**

I. Introduction

The reconfigurable System-on-Chip (RSoC) architectures combine processors with an FPGA fabric on a single die. Such single chip solution enables to reduce power consumption by effective application partitioning and/or to reduce the overall system's price significantly. Recent representatives are Xilinx Zynq All Programmable SoC [1] or Altera Cyclone V SoC [2]. Both chips consist of a dual-core ARM Cortex-A9 MPCore and an FPGA fabric build upon its producer's latest technology. The processor part of the system is interconnected with the FPGA part using several high and low performance ports based on the AMBA AXI [4]. A very useful feature is also the availability of different built-in peripherals (e. g. memory interface, Ethernet, SPI, USB, SD/SDIO, CAN, and others) that are usually missing in the standard FPGA as a hard-core.

There are several steps that must be taken when designing for similar complex architectures. At least, an application software and an FPGA accelerator (or peripheral) must be developed. There exist a plenty of different operating systems and toolchains for the software development on ARM-based processors. Also many libraries and standard drivers can be used for this part of the RSoC. FPGA firmware development can be particularly simplified by high level synthesis process (e. g. developing FPGA application by means of standard programming languages – C/C++). It can also benefit from publicly or commercially available IP cores.

For every application, accelerators and software must be integrated together. For example, a low-level software interface (a new Linux driver) must be written, DMA engine needs to be instantiated and properly configured to set up transfers in-between FPGA part and the processor part (memory) of the RSoC, etc. Basic drivers and existing hardware IP blocks can support the user in developing his application. However, integration work can lead to many critical problems as misconfiguration or misconnection is likely to happen. Moreover, this surrounding infrastructure needs to be tested and verified. To help developers with such issues, the new RSoC Framework is introduced and demonstrated here. The framework provides a customizable FPGA bridge (firmware part) and a corresponding set of generic OS drivers (software part). As a result, each application can profit from this unified environment that covers both the integration and testing. No such universal framework exists.

The rest of the paper is organized as follows. Section II describes our RSoC Framework, its basic parts and functionality. Simple video processing demo and other use-cases build upon this framework are shown in Sections III and IV. Finally, Section V gives conclusions and discusses our future work.

II. RSoC Framework

The RSoC Framework provides a middleware layer to support easy and fast prototyping of applications. It hides specifics of the underlying hardware platform (e. g. Xilinx Zynq All Programmable SoC [1], Altera Cyclone V SoC [2]). Developers are abstracted from several platform specific details that are completely solved by the framework. There are two main components of the framework: **RSoC Bridge** (firmware) and **RSoC Driver** (software). Both parts are shown in Fig. 1.

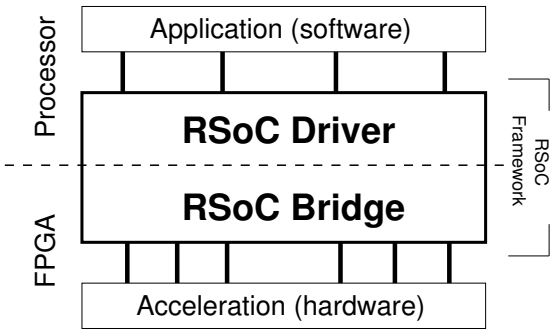

Fig. 1. RSoC Framework basic structure

A. Software part

The RSoC Driver is a driver for a Linux-based operating system that enables programs to exchange data with corresponding FPGA accelerators using the well-known Linux interfaces: `open`/`close`, `write`, `read`, `mmap`, and `ioctl`. RTOS is currently not supported but it is planned in near future. Each accelerator is represented by an instance of a character driver [3] and is accessible through a device node like `/dev/acc0`. A `write` or a `read` operation on such device node executes a DMA transfer between the memory and the corresponding accelerator in the firmware. Writing and reading of the configuration registers is also possible (implemented by the AXI-Lite bus [4]).

B. Firmware part

The RSoC Bridge is a highly configurable FPGA IP core. It integrates FPGA accelerators with the rest of the system by AXI-Stream buses and an optional AXI-Lite bus [4]. Based on its configuration, the RSoC Bridge generates an internal bus system, DMA engines and other necessary glue logic to connect user accelerators into the system. The RSoC Driver determines the RSoC Bridge's configuration at runtime and initializes itself accordingly. Several configurations are prepared and tested for common cases.

III. Video application demo

To demonstrate the RSoC Framework, a simple video application was designed. It was implemented on the Enclustra's Mars Starter Evaluation Base Board [5] with Mars ZX3 module. However, any other hardware board with the appropriate interfaces can be used as well.

The demonstration application is a video filter implemented as a software application with an optional acceleration in the FPGA part. The application receives video data from an input interface. Then it performs the filtering (median, sobel, etc.). The result is sent to an output. The I/O interfaces can be HDMI, filesystem, network video streams, or others (depending on the board). These interfaces can be implemented using existing libraries and drivers on top of the operating system.

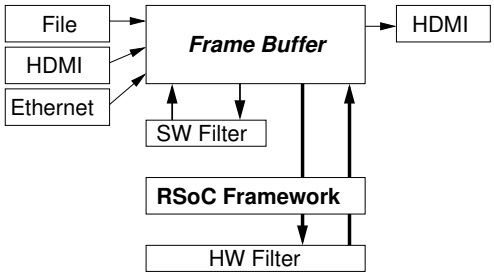

Fig. 2. Video application based on RSoC Framework

In case of high-quality video streams, the ARM Cortex-A9 is not able to process such a big amount of data. The hardware acceleration in the FPGA can solve this issue effectively. With the available tools, IP cores and application notes, the FPGA developer needs to implement the accelerator. Verification of the accelerator must be performed. Then the accelerator can be connected using a DMA engine to the selected memory interface available on the chip. Both the accelerator and the engine are connected to an available configuration bus and an address space is assigned to them. Verification of this complete system must be performed as well. The software developer implements a low-level OS driver that determines the corresponding physical addresses and controls the DMA engine. This require experience and knowledge about the Linux Kernel. The driver should be verified that it works correctly. Note that debugging of such drivers is a difficult task in general. Finally, the software application must be modified to access the driver instead of performing the computation itself.

The RSoC Framework addresses the above mentioned issues. The FPGA developer writes the accelerator and verifies its function. This step is same as without our framework. But then the accelerator can be connected to the RSoC Bridge. The verification environment of the RSoC Framework can be used to verify that the accelerator communicates in the expected way with the bridge. The application only needs to access the RSoC Driver instead of the actual computation. Again, the contained testing/verification environment can be utilized to assure that the application calls the RSoC Driver as expected. The RSoC Driver and RSoC Bridge are already tested and verified to work correctly together.

IV. Other applications

It is obvious that the same simplification of application development using RSoC Framework is applicable in other areas as well. Consider, for example, reliable network wire-speed applications or applications using cryptographic algorithms.

The RSoC Framework can provide support for application of partial and dynamic reconfiguration out of the box. Any accelerator connected to the RSoC Bridge (when synthesized accordingly) can be dynamically loaded into the FPGA part. A software controller can decide to accelerate a part of a computation based on the current processor load.

V. Conclusion

This work presents new RSoC Framework for reconfigurable SoCs. The efficiency of rapid application prototyping has been demonstrated by different use-cases with insight to video application. It was shown that building different applications that utilize the RSoC Framework is straightforward. More information can be found at *www.rsoc-framework.eu*.

The future work will be focused on porting our framework and demonstration applications to other architectures. However this process will be simplified as our RSoC Framework has been designed to support similar architectures.

Acknowledgment

This work has been supported by the Sec6Net project *Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet* no. VG20102015022.

References

[1] Xilinx, Inc. Zynq-7000 All Programmable SoC: Technical Reference Manual [online]. 2012. 1707 p. UG585.

[2] Altera, Corp. Cyclone V Device Handbook [online]. 2012. 1096 p.

[3] Corbet, J.—Rubini, A.—Kroah-Hartman, G. Linux Device Drivers. Chapter 3. Third edition. O'Reilly Media, February 2005. ISBN: 0-596-00590-3.

[4] Xilinx, Inc. AXI Reference Guide [online]. 2012. 132 p. UG761.

[5] Enclustra, GmbH. Mars Starter Evaulation Base Board [online]. 2013. 2 p. Product Brief.